Pre-sorted Forward-Backward NB-LDPC Check Node Architecture

Hassan Harb, Cédric Marchand,
Laura Conde-Canencia and Emmanuel Boutillon
Lab-STICC, CNRS UMR 6285
Université de Bretagne Sud, Lorient, France
Email: emmanuel.boutillon@univ-ubs.fr

Ali Al Ghouwayel
CCE Department
Lebanese International University (LIU), Beirut, Lebanon.
Email: ali.ghouwayel@liu.edu.lb

***Abstract*—This paper deals with reduced-complexity NB-LDPC check node implementation based on the Extended Min-Sum algorithm. We propose to apply a recently introduced pre-sorting technique to the forward-backward architecture. The pre-sorting of the check node inputs allows for significant complexity reduction. Simulation and synthesis results showed that this approach does not introduce any performance loss and can lead to significant area reduction in FPGA implementations (up to 54% for high check node degrees).**

***Index Terms*—NB-LDPC, Check Node, Forward-backward.**

I. Introduction

Because of their near-channel-capacity performance, Low-Density Parity-Check (LDPC) codes [1] have been adopted for a wide range of standards (WiMAX, WiFi, DVB-C2, DVB-S2X, DVB-T2). However, this performance is mainly obtained for long codeword lengths and LDPC start to show their weakness when considering short and moderate codeword lengths. The last decade witnessed a great deal of research effort devoted to the extended version of LDPC codes defined over (GF(q), $q > 2$). These codes called Non-Binary (NB) LDPC codes have shown strong potential in error correction capability with moderate and short codeword lengths [2]. NB-LDPC retain the benefits of steep waterfall region (typical of convolutional turbo-codes) and low error floor (typical of binary LDPC). Compared to their binary counterparts, NB-LDPC codes generally present higher girths, which leads to better decoding performance. Moreover, the NB nature of such NB-LDPC codes makes them suitable for high-spectral-efficiency modulation schemes where the constellation symbols are directly mapped to GF(q) symbols. This mapping bypasses the marginalization process of binary LDPC codes that causes information loss. These characteristics place NB-LDPC codes as serious competitors of classical binary LDPC and Turbo-Codes in future wireless communication and digital video broadcasting standards. However, NB-LDPC codes suffer from high decoding complexity. In the NB decoder each message exchanged between processing nodes is an array of values, each value corresponding to a GF element. From an implementation point of view, this leads to a highly increased complexity compared to binary LDPC.

The direct application of the Belief Propagation (BP) algorithm to NB-LDPC codes leads to a computational complexity dominated by $\mathcal{O}(d_c \cdot q^2)$ [2] for each check node update, which becomes prohibitive when considering values of $q > 16$. An important effort has thus been dedicated to develop reduced-complexity algorithms for NB-LDPC decoding. The authors in [3] proposed to perform the BP algorithm in the logarithmic domain. In [4] an FFT-Based BP decoding algorithm was proposed. However, although these algorithms considerably reduce the computational complexity of the decoding process, they are still far from being considered for hardware implementation. This implementation became feasible with the introduction of the Extended Min-Sum (EMS) [5] and the Min-Max [6] algorithms.

The EMS algorithm [5] is based on a generalization of the Min-Sum algorithm and it has the advantage of performing only additions while truncating the size of the messages from q to n_m ($n_m \ll q$). This sub-optimality introduces a performance degradation that is compensated by a correction factor that can be optimized so that the EMS algorithm can approach, or even in some cases slightly outperform, the FFT-based BP decoder.

The bottleneck of the EMS algorithm is considered to be at the Check Node (CN) processing step. Two approaches have been proposed for EMS CN implementation: the Forward-Backward CN (FB-CN) [3] and, more recently, the Syndrome architecture (SB-CN) [7]. Unlike the serial processing of the FB-CN, the SB-CN allows for parallel CN design. However the complexity of the SB-CN architecture is given by the number of computed syndromes which is in the order of $\mathcal{O}(d_c^2)$ thus reducing its interest for high d_c values. A powerful original technique to reduce the number of syndromes has been recently presented in [8]. This so-called pre-sorting technique reorders the d_c CN inputs in order to simplify the CN architecture without affecting performance.

In this paper we investigate the application of the pre-sorting technique to the FB-CN approach to obtain significant complexity reduction of the NB-LDPC CN implementation. The sorting of the d_c CN inputs prior to the CN processing modifies the behaviour of the Elementary Check Nodes (ECN). This behaviour is analyzed through statistics and, from this analysis, the architecture of each ECN is simplified leading to a design that uses the minimum number of hardware resources that guarantee a correct processing. Among the different

reduced-complexity architectures for EMS ECN processing (see [9] [10] [11]), we considered the S-Bubble which presents the better performance/complexity trade-off [11].

This paper is organised as follows: section II reviews NB-LDPC EMS CN processing and the FB implementation. Section III presents the pre-sorting technique and its application to the FB-CN architecture. Then, Section IV describes the pre-sorted FB-CN (P-FB) with the simplified ECNs. The implementation and simulation results are presented and discussed in Section V. Finally, Section VI concludes the paper.

II. NB-LDPC Check Node processing

A. Check Node and LLR description

In the Tanner bi-partite graph representation of NB-LDPC codes, a CN processor is connected to d_c Variable Nodes (VN) (Fig. 1). The CN processor receives d_c variable-to-node messages and, after processing, it generates d_c node-to-variable messages.

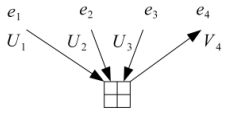

Fig. 1. Input and output messages in a Check Node.

Let us define a CN equation of degree d_c in GF(q) as $e_1 \oplus e_2 \oplus e_3 \oplus \ldots \oplus e_{d_c} = 0$, where the operator $\oplus$ is used to represent the addition in GF(q). Each input e_i can take q values. The *a priori* information about variable e is the discrete probability distribution $P(e = x)$, $x \in$ GF(q). Each element of the probability distribution E associated to e can be expressed in the logarithmic domain as the Log Likelihood Ratio (LLR) denoted by $e^+(x)$:

$$e^+(x) = -\log\left(\frac{P(e=x)}{P(e=\bar{x})}\right) \tag{1}$$

where $\bar{x}$ is the hard decision on e obtained by taking the most probable GF symbol, i.e. $\bar{x} = \arg\max_{x \in \text{GF}(q)} P(e = x)$.

By definition of the LLR, we have: $e^+(\bar{x}) = 0$ and $\forall x \in$ GF(q), $e^+(x) \geq 0$. The distribution (or message) E associated to e is thus $E = \{e^+(x)\}_{x \in \text{GF}(q)}$.

B. The EMS algorithm

As already introduced, the EMS algorithm truncates the q values to only the n_m most reliable ones, with $n_m << q$. Thus, each input U of a CN is a list $\{U[j]\}_{j=0\ldots n_m-1}$ of couples, with $U[j] = (U^+[j], U^{\oplus}[j])$, where $U^+[j]$ designates the j^{th} smallest LLR value of E and $U^{\oplus}[j]$ is its associated GF element, i.e., $e^+(U^{\oplus}[j]) = U^+[j]$. Note also that $U^+[0] = 0$, $U^{\oplus}[0] = \bar{x}$, and that $j \leq j' \Rightarrow U^+[j] \leq U^+[j']$.

The Min-Sum algorithm equation that defines the LLR value of the GF symbol x for the i^{th} output is:

$$v_i^+(x) = \min\left\{ \sum_{i'=1, i' \neq i}^{d_c} e_{i'}^+(x_{i'}) \mid \bigoplus_{i'=1, i' \neq i}^{d_c} x_{i'} = x \right\}, \tag{2}$$

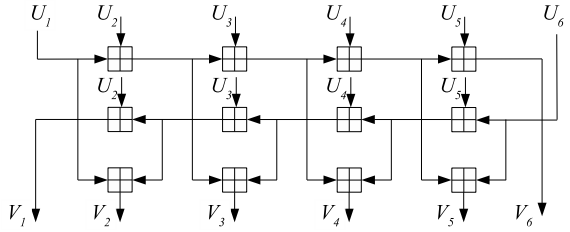

Fig. 2. FB-CN processing with $d_c = 6$.

where $x_{i'} \in$ GF(q) for $i' = 1, \ldots, d_c$, $i' \neq i$. From this equation, we can simplify for the EMS by replacing $x_{i'} \in$ GF(q) by $U_i^{\oplus} = \{U_i^{\oplus}[j]\}_{j=0,1,\ldots,n_m-1}$, as only n_m values are considered:

$$v_i^+(x) = \min\left\{ \sum_{i'=1, i' \neq i}^{d_c} U_{i'}^+[j_{i'}] \mid \bigoplus_{i'=1, i' \neq i}^{d_c} U_{i'}^{\oplus}[j_{i'}] = x \right\}, \tag{3}$$

where $j_{i'} \in \{0, 1, \ldots n_m - 1\}$ for $i' = 1, 2, \ldots d_c$, $i' \neq i$. Then, the $v_i^+(x)$ values are sorted in increasing order and only the first n_m smallest values constitute the output vector V_i. Note that $V_i^+[0] = 0$ (in fact, $j_{i'} = 0 \Rightarrow U_{i'}^+[j_{i'}] = 0$ and thus, at least one term in (3) is the addition of zero values).

C. Forward-Backward CN processing

The FB-CN algorithm divides the CN processing in three layers: forward layer, backward layer and merge layer. Each layer is composed of $d_c - 2$ Elementary CNs (ECNs). An ECN is a block that processes a single output C as a function of two inputs A and B. Fig. 2 shows the resulting structure for a CN with $d_c = 6$ and $3 \times (d_c - 2) = 12$ ECNs. Intermediate results of the ECNs are reused in the last stages and avoid re-computations [9].

The ECN processing can be performed as follows: for each couple $(a, b) \in \{0, 1, \ldots, n_m - 1\}^2$, the output couples $C_{a,b} = (C_{a,b}^+, C_{a,b}^{\oplus})$ are computed as:

$$C_{a,b}^+ = A^+[a] + B^+[b],\ C_{a,b}^{\oplus} = A^{\oplus}[a] \oplus B^{\oplus}[b] \tag{4}$$

Then, the couples $C_{a,b}$ are sorted in increasing order of $C_{a,b}^+$. The output vector C corresponds to the first n_m couples with the n_m smallest values of $C_{a,b}^+$. The candidate values to be output are called *bubbles*. Note that, since $A^+[0] = 0$ and $B^+[0] = 0$, then the first bubble in C is $C(0) = C_{0,0} = (0, A^{\oplus}[0] \oplus B^{\oplus}[0])$. Moreover, since A and B are ordered in increasing order of their LLR, then $a' \leq a$, $b' \leq b \Rightarrow C_{a',b'}^+ \leq C_{a,b}^+$. In other words, there are at least $(a+1)(b+1) - 1$ LLRs lower than $C_{a,b}^+$. Since only the first n_m smallest LLRs are output, all bubbles of index (a, b) verifying $(a+1)(b+1) > n_m$ are directly excluded. The total number of considered bubbles is thus a function of n_m and will be denoted $\phi(n_m)$.

In [10] the authors showed that, in practice, the role of an EMS ECN processor is to select the n_m most reliable symbols in a 2D matrix T_Σ, where $T_\Sigma(i, j) = A(i) + B(j)\ \ \forall (i, j) \in$

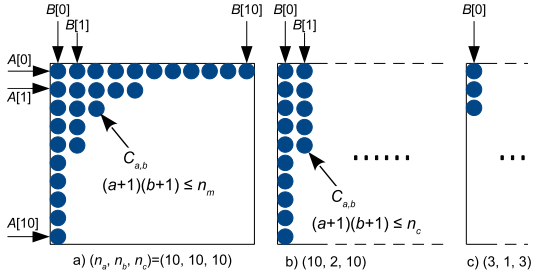

Fig. 3. S-bubble check and generalized S-bubble with $n_m = 10$.

$[1, n_m - 1]^2$. Different architectures have then been proposed [10], [11] based on the simplification of the symbol selection.

In [9] and [11], the ECN is simplified by excluding the bubble with both a and b greater than 1. In this study, we now propose to generalize the variation range of a, b and c from 0 to $n_m - 1$ to respectively 0 to $n_a - 1$, 0 to $n_b - 1$ and 0 to $n_c - 1$. In Fig. 3.(a), the possible combinations of a and b are represented by circles and are referred as *potential bubbles*. For example, for $n_m = 10$, $\phi(10) = 28$ in Fig. 3.(a). Different scenarios for $(n_a, n_b, n_c) = (10, 10, 10), (10, 2, 10)$ and $(3, 1, 3)$ are considered in Fig. 3 where the *potential bubbles* are represented.

III. Input Message Pre-Sorting

The idea of the input pre-sorting is to polarize the statistics of d_c variable-to-check messages by sorting them according to the reliability of the hard decision input, i.e., the probability $P(e_i = U_i^{\oplus}[0])$, $i = 1, 2, \ldots, d_c$. The reason for this approach is that many bubbles in the ECN are very unlikely to contribute to the output, suppressing them does not affect performance but can lead to architectural simplifications.

A. Pre-sorting technique to polarize the input messages

Considering Eq. (1) and knowing that $\sum_{x \in \mathrm{GF}} P(e_i = x) = 1$, the probability $P(e_i = U_i^{\oplus}[0])$, $i = 1, 2, \ldots, d_c$ can be expressed as:

$$P(e_i = U_i^{\oplus}[0]) = \frac{1}{\sum_{j=0}^{q-1} e^{-U_i^+[j]}}. \quad (5)$$

Note that in Eq. 5, the values of $U_i^+[j]$ for $j \ge n_m$ are equal to $U_i^+[n_m - 1] + O$, where O is a constant offset value, as detailed in [12]. Since $U_i^+[0] = 0$ and, for $j > 2$, $U_i^+[1] \le U_i^+[j]$, then $P(e_i = U_i^{\oplus}[0])$ can be approximated by:

$$P(e_i = U_i^{\oplus}[0]) \approx \frac{1}{1 + e^{-U_i^+[1]}}. \quad (6)$$

In other words, the higher the value of $U_i^+[1]$, the higher $P(e_i = U_i^{\oplus}[0])$ is. From this, we can state that the pre-sorting step is performed according to vector $U^1 = (U_1^+[1], U_2^+[1], \ldots, U_{d_c}^+[1])$ as described by the Algorithm 1. As shown in the example of Fig. 4 for n_m = 5 and d_c = 4, only a reduced number of values in the sorted vectors $\{U_i'\}_{i=1,2,\ldots,d_c}$ are considered as inputs to the EMS CN block.

Input The d_c input message $\{U_i\}_{i=1,2,\ldots,d_c}$.
Step 1: Extract vector $U^1 = (U_1^+[1], U_2^+[1], \ldots, U_{d_c}^+[1])$ Sort U^1 in ascending order to generate U'^1.
return permutation $\pi = (\pi(1), \ldots, \pi(d_c))$ associated to the sorting process: $U'^1(i) = U^1(\pi(i))$, $i = 1, 2 \ldots d_c$.
Step 2: Permute input vectors using the permutation π: for $i = 1, 2, \ldots, d_c$, $U_i' = U_{\pi(i)}$
Step 3: Perform the CN process with input vectors $\{U_i'\}_{i=1,2,\ldots,d_c}$ to generate output vectors $\{V_i'\}_{i=1,2,\ldots,d_c}$.
Step 4: Permute output vector using the inverse permutation π^{-1}: for $i = 1, 2, \ldots, d_c$, $V_{\pi(i)} = V_i'$

Algorithm 1: Pre-sorting principle

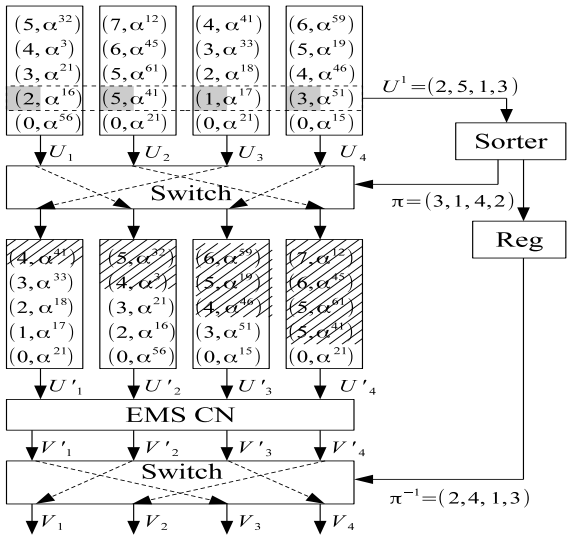

Fig. 4. Pre-sorting principle

This observation motivated the original approach described in the following subsection.

B. Pre-sorting with the FB-CN algorithm

A statistical study of the behaviour of the bubbles in the input vectors $\{U_i'\}_{i=1,2,\ldots,d_c}$ allow us to predict for each ECN the hardware resources that can be simplified without affecting the global performance of the CN processing. This study is performed though the observation of each ECN processing during the Monte-Carlo simulation of a (576, 480) GF(64)-LDPC code at SNR = 3.5 dB over more than ten thousand decoded frames.

To be specific, Fig. 5 represents a S-Bubble Check FB-CN with $d_c = 12$ and three layers of $d_c - 2 = 10$ ECNs each. The points (inside or outside each small square) represent the positions of the processed bubbles with the S-Bubble architecture [11], which are a total of $b = 1680$. The small squares represent the positions of the bubbles that contribute to an output after applying the pre-sorting technique. They represent a number of $b^o = 648$ bubbles, i.e. about $0.4 \times b$. Consider for example ECN B10 in the Backward layer: the

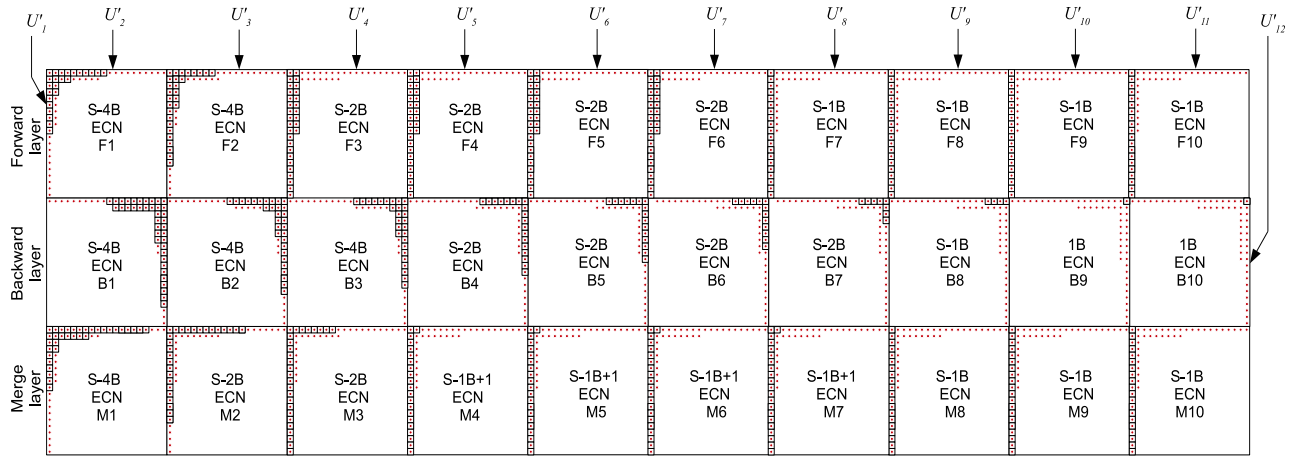

Fig. 5. Matrix representation of a S-Bubble Check FB-CN with $d_c = 12$ and $n_m = 20$. The $b = 1680$ red circles represent the bubbles in the original FB-CN algorithm. The squares represent the remaining $b^o = 648$ bubbles after the pruning process in the S-FB algorithm.

processing of only one bubble is necessary and implementing a S-Bubble architecture at this ECN clearly implies a waste of resources. The idea is then to implement for each ECN the most simplified architecture that guarantees a correct ECN processing as detailed in the following section. Please also note that the pre-sorting technique requires extra hardware blocks compared to the classical *not sorted* CN architecture: a d_c-input vector sorter and two permutation networks (or switches). We will also show in section V that the area cost of this extra hardware is compensated with the ECN simplifications, leading to an optimised global CN implementation.

IV. Proposed FB-CN Architecture

This section first describes the elementary blocks constituting the proposed FB-CN architecture: sorter, switch and simplified ECNs. Then, the global CN architectures for different d_c values are presented.

A. Sorter

Several sorting algorithms have been proposed in the literature based on serial [13] and parallel approaches [14]. The selection of the most suitable sorter architecture is based on two main criteria: hardware complexity and speed performance. The architecture of the sorter we implemented is a semi-parallel architecture based on the algorithm proposed in [15]. The architecture is composed of $d_c/2$ stages where each stage contains $d_c - 1$ comparator-swap blocks as shown in Fig. 6. Since the processing time of the FB-CN processor will be greater than the sorting time, we have implemented only one stage that will be running $d_c/2$ times in order to sort an input vector of size d_c and to generate the permutation order vector π (see Fig. 4). The latency of this sorter architecture is $d_c/2$ cycles and it constitutes a good trade-off between complexity and performance.

B. Switch

The Switch block receives the d_c inputs $\{U_i\}_{i=1,2,\ldots,d_c}$ and permutes them based on the permutation vector π received from the Sorter. This Switch is composed of d_c multiplexers of size d_c-to-1, as shown in Fig. 6.

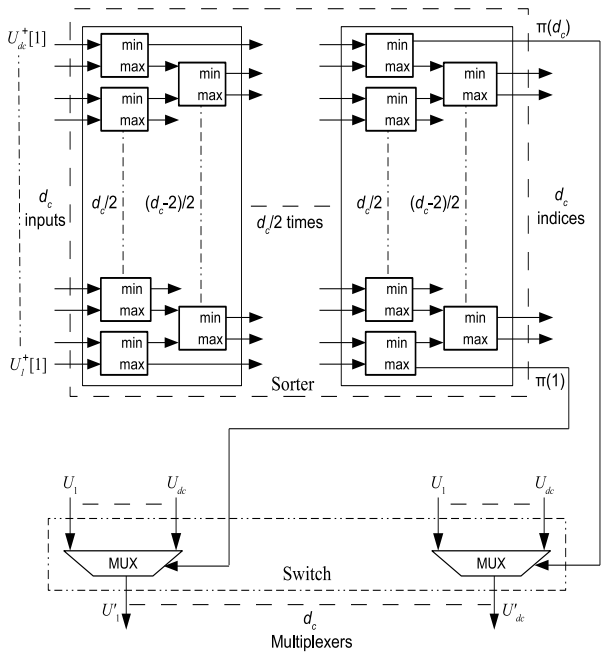

Fig. 6. Architecture of the Sorter and Switch blocks. The Sorter architecture follows [15].

C. Simplified ECNs

As previously mentioned in section III-B the hardware resources of each ECN can be reduced without affecting performance. Five different simplified ECN architectures can be considered:

1) S-4B: This ECN architecture known as S-bubble ECN is described in [11] where four bubbles are compared per clock cycle. It is composed of 4 FIFO blocks, six comparators, four arithmetic adders and four modulo-2 adders implemented using XOR gates.

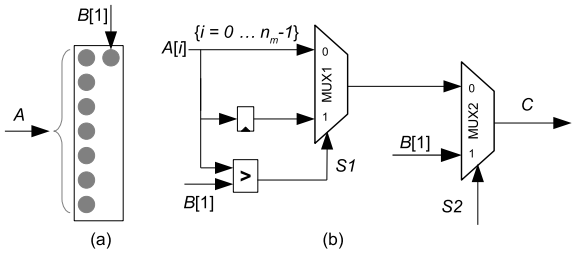

Fig. 7. S-1B+1 ECN and its architecture

TABLE I
NUMBER OF ECN SCHEMES FOR DIFFERENT d_c VALUES

S-FB	S-4B	S-2B	S-1B	S-1B+1	1B
d_c = 6	5	7	-	-	-
d_c = 8	9	5	2	1	1
d_c = 12	6	10	8	4	2
d_c = 20	12	7	24	1	10

2) S-2B: It is based on the S-bubble ECN but composed of only the first row and first column of matrix T_Σ. Thus, only two bubbles are compared per clock cycle, two FIFO blocks are needed with only one comparator and two modulo-2 adders.

3) S-1B: This vector ECN generates the output C as: $C^{+}_{a,0} = A^{+}[a]$, $C^{\oplus}_{a,0} = A^{\oplus}[a] \oplus B^{\oplus}[0]$. Note that vectors A and B can be exchanged depending on the distribution of the bubbles.

4) S-1B+1: The bubbles considered in this ECN processing are shown in Fig. 7.a and the architecture in Fig. 7.b. It is composed of a comparator, two 2-to-1 multiplexers and a single register. The control signal S_1 is initially 0 and then set to 1 for all the following cycles if and only if $A^{+}[i] > B^{+}[1]$, i=1, 2, ... $n_m - 1$ and $C^{\oplus}[i,0] \neq C^{\oplus}[0,1]$. The control signal S_2 is also initialized to 0 and keeps this value while $A^{+}[i] < B^{+}[1]$. It will be turned to 1 for only one cycle when $A^{+}[i] > B^{+}[1]$ and $C^{\oplus}[0,1]$ is different to all the symbols $C^{\oplus}[j,0]$, $j < i$ already output.

5) 1B: This ECN considers a single bubble where the output is the most reliable element $C^{\oplus}_{0,0} = A^{\oplus}[0] \oplus B^{\oplus}[0]$.

D. ECNs simplifications for global CN with different d_c values

The statistical analysis and architectural ECN simplifications were performed for $d_c = 6, 8, 12$ and 20, i.e. coding rates 2/3, 3/4, 5/6 and 9/10, respectively. For $d_c = 12$, Fig. 5 details the ECN architecture retained for each ECN. Table I presents the number of implementations of each kind of ECN architecture in the global CN as a function of the d_c values. From these results we can predict significant potential area gains specially for high d_c values: for example, for $d_c = 20$ the S-Bubble architecture will be replaced 10 times by the 1B architecture.

V. IMPLEMENTATION AND SIMULATION RESULTS

To quantify the interest of the pre-sorting technique in FB-CN architectures we implemented the different architectural designs on a FPGA device. We also show simulations results of the new approach without performance loss.

TABLE II
POST SYNTHESIS RESULTS FOR DIFFERENT ECN SCHEMES ON A XILINX VIRTEX 6 FPGA.

ECN	Number of occupied slices	Frequency (MHz)	Latency (cycles)
1B	7	714	1
S-1B	17	714	1
S-1B+1	35	349	1
S-2B	82	334	2
S-4B	138	269	2

TABLE III
POST-SYNTHESIS RESULTS FOR THE FB-CNS WITH (P-FB) AND WITHOUT (S-FB) PRE-SORTING ON A XILINX FPGA DEVICE

FB-CN		Nb. of occupied slices				
d_c	Case	Sorter	Switch	CN	Total	Gain(%)
6	S-FB	0	0	1,617	1,617	5
	P-FB	50	93	1,268	1,532	
8	S-FB	0	0	2,481	2,481	17
	P-FB	77	142	1,701	2,061	
12	S-FB	0	0	4,666	4,666	43
	P-FB	160	283	1,858	2,653	
20	S-FB	0	0	6,519	6,519	54
	P-FB	386	495	1,232	2,955	

A. Implementation Results

We considered the Xilinx VIRTEX 6, xc6vlx240t-2ff1156 FPGA device to obtain synthesis results. The five ECN architectures were synthesized to obtain the results presented in Table II. The LLR and GF values are quantified on 6 bits. The 1B and S-1B ECNs have negligible complexity and a maximum frequency of 714 MHz. Also, the S-1B+1 and S-2B ECNs have reduced complexity compared to S-4B and can operate at higher frequencies.

Table III summarizes the overall complexity of the FB-CN for different considered d_c values. Please note that "S-FB" stands for the S-Bubble CN implementation and that "P-FB" stands for the presorting approach proposed in this paper. The proposed architecture leads to a global CN complexity reduction of 5% for $d_c = 6$, 43% for $d_c = 12$ and 54% for $d_c = 20$, compared to the state-of-the-art S-FB architecture.

Table III also shows the synthesis results of the Sorter and Switch blocks. These extra blocks (1 Sorter and 2 Switches) of the pre-sorting step constitute about 30% (46%) of the total area for $d_c = 12$ (resp. $d_c = 20$), but the ECN architectural simplifications compensate for this and a global gain of 43% (resp. 54%) is obtained.

Even if the implementation of the variable node is out of the scope of this article, let us note that significant area reduction is expected as the number of sorted values to compute for CN input messages is reduced. For $d_c = 12$, the $n_m = 20$ values (for each message) is reduced to a maximum of 10 for U'_1 and a minimum of 1 for U'_{12}, as shown in Fig. 5.

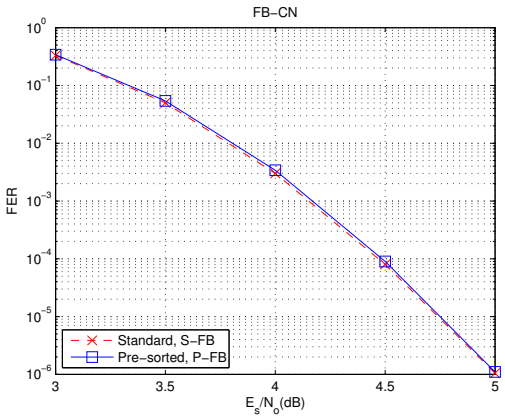

Fig. 8. Simulation results of NB-LDPC decoding algorithms for (576, 480) code over GF(64) and $d_c = 12$ under AWGN channel.

B. Simulation Results

Finally we present bit-true Monte-Carlo simulation results over the AWGN channel with a BPSK modulation scheme. Extrinsic and intrinsic LLR messages are quantified on 6 bits and a-posteriori LLRs on 8 bits. The two scenarios presented in Fig. 5 are considered: 1. the S-FB which correspond to the state of the art S-Bubble approach without pre-sorting; 2. the P-FB which corresponds to the approach with pre-sorting and ECN simplifications;

Fig. 8 shows the simulations results for a (576, 480), $d_c = 12$ GF(64)-LDPC code. The pre-sorting technique shows negligible performance degradation while implementation results in Table III shows 43 % complexity reduction with $d_c = 12$.

VI. Conclusion

This paper proposed an efficient forward-backward CN architecture based on a pre-sorting technique of the input messages. The pre-sorting technique significantly reduces the number of the values that contribute to the CN processing. Then, the architectural design can be simplified for each elementary architecture leading to a reduced-area implementation, specially for high d_c values. To be specific, area reductions of 43% and 54% were obtained for CN degrees of $d_c = 12$ and 20, respectively. Simulation results showed that this complexity reduction does not entail any performance loss. Finally, this work can also lead to simplified variable node implementations.

References

[1] R. Gallager, "Low-density parity-check codes," Ph.D. dissertation, Cambridge, 1963.

[2] M. Davey and D. MacKay, "Low-density parity check codes over GF(q)," *Communications Letters, IEEE*, vol. 2, no. 6, pp. 165–167, June 1998.

[3] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Communications, 2004 IEEE International Conference on*, vol. 2, June 2004, pp. 772–776.

[4] D. MacKay and M. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *In Codes, Systems and Graphical Models*. Springer-Verlag, 1999, pp. 113–190.

[5] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. on Commun.*, vol. 55, pp. 633–643, April 2007.

[6] V. Savin, "Min-max decoding for non binary LDPC codes," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, July 2008, pp. 960–964.

[7] P. Schlafer, N. Wehn, M. Alles, T. Lehnigk-Emden, and E. Boutillon, "Syndrome based check node processing of high order NB-LDPC decoders," in *Telecommunications (ICT), 2015 22nd International Conference on*, April 2015, pp. 156–162.

[8] C. Marchand and E. Boutillon, "NB-LDPC check node with pre-sorted input," in *9th International Symposium on Turbo Codes & Iterative Information Processing*, September 2016.

[9] E. Boutillon, L. Conde-Canencia, and A. A. Ghouwayel, "Design of a GF(64)-LDPC decoder based on the EMS algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 10, pp. 2644–2656, Oct 2013.

[10] E. Boutillon and L. Conde-Canencia, "Bubble check: a simplified algorithm for elementary check node processing in extended Min-Sum non-binary LDPC decoders," *Electronics Letters*, vol. 46, no. 9, pp. 633–634, 2010.

[11] O. Abassi, L. Conde-Canencia, A. A. Ghouwayel, and E. Boutillon, "A novel architecture for elementary check node processing in non-binary LDPC decoders," *Transactions on Circuits and Systems II: Express Briefs, accepted for publication*, 2016.

[12] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Transactions on Communications*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[13] D. Koch and J. Torresen, "FPGASort: A high performance sorting architecture exploiting run-time reconfiguration on FPGAs for large problem sorting," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2011, pp. 45–54.

[14] A. Farmahini-Farahani, H. J. D. III, M. J. Schulte, and K. Compton, "Modular design of high-throughput, low-latency sorting units," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1389–1402, July 2013.

[15] J. Martinez, R. Cumplido, and C. Feregrino, "An FPGA-based parallel sorting architecture for the burrows wheeler transform," in *2005 Int. Conf. on Reconfigurable Computing and FPGAs*, Sept. 2005.